# odfWeave and Data Analysis Platforms

Max Kuhn

max.kuhn@pfizer.com

Pfizer Global R&D
Nonclinical Statistics
Groton, CT

# Outline

- Data analysis applications
- The need for documentation
- Sweave
- Sweave and the Open Document Format

# Data Analysis Systems

In a large organization, statisticians are greatly outnumbered.

Clearly, some statistical techniques cannot be automated for a variety of reasons. Good candidates for automation may include analyses where:

- a significant amount of manual input is not required during the computations
- the same computations will be conducted a large number of times
- the experimental design or data structures are well–defined and stable
- users can be trained to understand the analysis technique, the output and other aspects of the analysis (such a model assumptions or when the technique is not appropriate).

Two applications that can be effectively automated are *in–silico* chemistry models and RNA expression profiling analysis.

# in–Silico Chemistry Models

aka Quantitative Structure Activity Relationships – QSAR

aka Quantitative Structure Property Relationships – QSPR

Here, we want to create a model that uses the molecular structure of compounds to predict some properties/activity of other compounds.

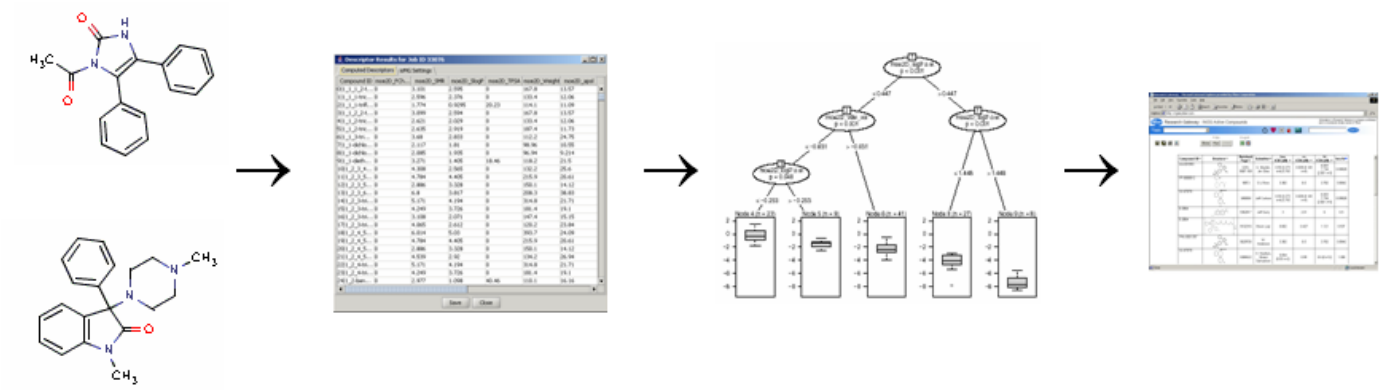
Examples of endpoints are:

- biological activity assay results,
- absorption, distribution, metabolism, excretion (ADME) endpoints
- toxicological assessments (QT elongation, etc)

# in–Silico Chemistry Models

Once an acceptable model is created, the prediction algorithm is published in internal systems and chemists can use it to predict other compounds.

In this way, new (or virtual) compounds and be screened or ranked to find good candidate compounds.

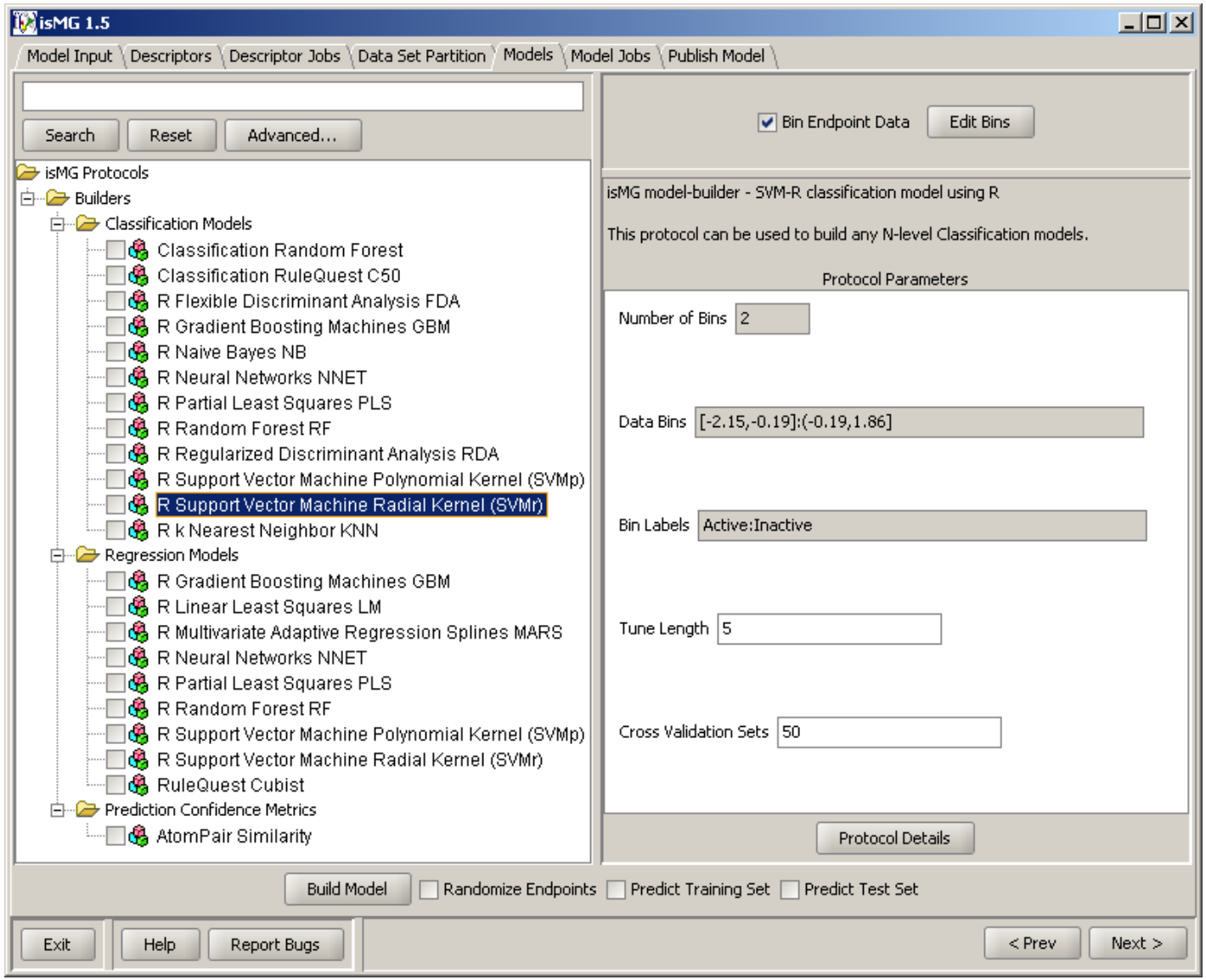


# in–Silico Chemistry Models

The in–silico Model Generator (isMG) is a Pfizer application/toolbox for building QSAR models.

The application is a client–side Java interface to Pfizer's computational grid (for R and other applications).

# The in–silico Model generator (isMG)

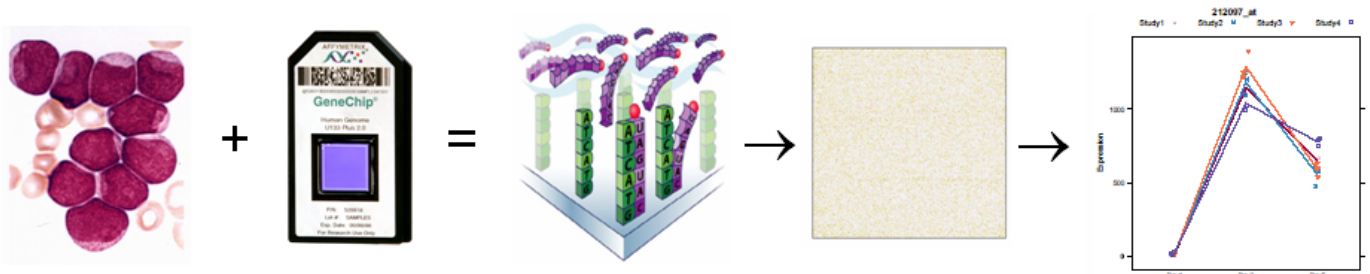


# RNA Expression Profiling

Microarrays allow the simultaneous measurement of RNA expression levels for thousands of gene targets. Common uses in drug discovery are

- discovering new drug targets
- biomarker selection for drug development
- understanding mechanism of action
- pathway analysis and biological modeling
- assessing toxicological ("off–target") effects

A typical experiment is designed to assess differential expression between groups of samples. This may allow scientists to understand the underlying genomic response to a stimulus.



# RNA Expression Profiling

The **Mi**croarray **D**ata **A**nalysis **S**ystem (Midas), is another Pfizer tool that can be used for assessing differential gene expression.

The basic data analysis process is:

1. sample annotation
2. probeset specification (focus probes, unsupervised filters)
3. data normalization
4. chip quality assessment
5. model/contrast specification
6. visualization and reporting of the results

The current Midas is a browser–based interface via Zope to R on the Pfizer grid. The application GUI is being transitioned to Java.

# Reporting the Results

For these examples, the end–users are primarily scientists in Discovery or Drug Safety.

In the case of expression profiling, the data analysis is the start of a larger process to understand complex biological functions. The results may be fed into additional software for pathway analysis or into biological network models.

For chemistry models, the computation chemists need to document their models with details on the compounds used, the descriptors, model performance and other application–specific details.

This demonstrates the need to collaboratively create written records of the model or analysis.

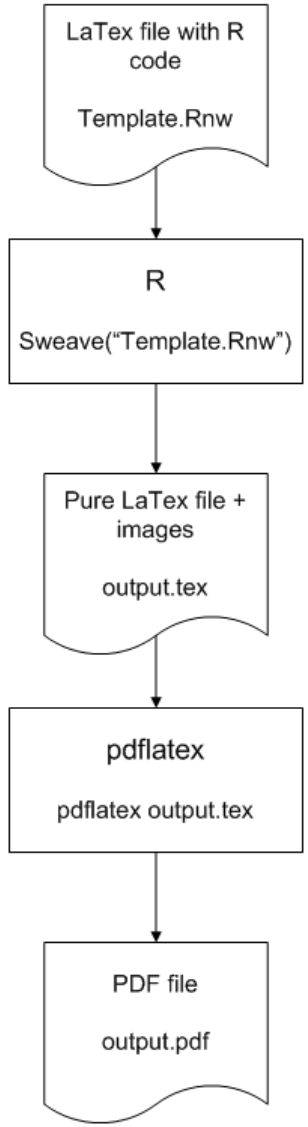
# Using Sweave for Generating Reports

Sweave is a process of embedding R code into other types of documents (like LaTeX or HTML) so that the results of the R code is intermixed with the rest of the document.

A template in LaTeX or HTML is written that contains R code via "code chunks".

In R, the Sweave function is called and the document is processed. R code is removed and the output of the code (LaTeX/HTML markup or image files) is put in its place.

In the case of LaTeX, pdflatex can be used to produce a PDF document



# Using Sweave for Generating Reports

Many of the outputs from isMG and Midas are contained in R data objects. From these files, relevant information can be extracted and used to document the details of the experiment or the results.

For example, in QSAR models, information can be extracted from the descriptor matrix about the frequencies of the types of molecular descriptors use in a model can be documented via in–line code chunks:

"There were \Sexpr{numDescr} descriptors used as inputs into the model building process, including \Sexpr{listString(descrFreq)} descriptors "

"There were 105 descriptors used as inputs into the model building process, including 44 GTN, 18 LCALC and 43 moe2D descriptors "

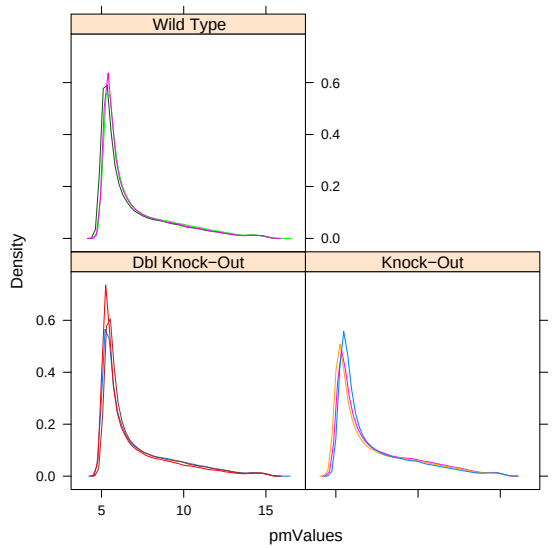
# Using Sweave for Generating Reports

Code chunks can also be added to the LaTeX markup to embed tables and images into the final document:

```
<<makeTable, echo = FALSE, results = tex>>=
latex(
   designFactors,
   file = "",
   caption = "",
   rowname = NULL,
   ctable = TRUE)
@
```

| Levels | N |
|---|---|
| Dbl Knock-Out | 3 |
| Knock-Out | 3 |
| Wild Type | 3 |

```
<<makePlot, echo = FALSE, fig = TRUE>>=
out <- densityplot(
   ~ pmValues|Group,
   data = tmp2,
   groups = File,
   plot.points = FALSE)
print(out)
@
```



# Report Format Issues

Currently, `Sweave` can easily create beautiful `PDF` or `HTML` documents.

Unfortunately, these formats cannot be easily edited, so the scientists have difficulties adding their subject–specific information about the results, as well as adding additional results (such as validation experiments).

Because of this, it would be a good idea to use document formats that are most familiar to our clients. Unfortunately, this means that we are locked into formats that are assessable using Microsoft Word.

There are some conversion applications, but our experience has not been favorable in converting `PDF`/`HTML` to `doc`/`rtf` formats.

# The Open Document Format (ODF)

ODF is a XML–based format that can be used to create text documents, spreadsheets and presentations. The format is open and non–proprietary. It was developed by the Organization for the Advancement of Structured Information Standards (OASIS) consortium. It is also an approved ISO standard.

Several open–source applications can be used to create or view these file, such as OpenOffice, NeoOffice, KWord and others. There are several add–in modules for MS office to import and export ODF files.

We'd like to be able to add R code to word–processing documents and "Sweave" them to embed the results:

- users who don't know (or want to know) markup languages can create reports with R output
- clients can more easily edit the documents

# The odfWeave Package

The odfWeave package extends Sweave to the Open Document Format.

An ODF file can be created using OpenOffice or other applications that contains Sweave tags, such as code chunks, that produce output. Paragraphs, bulleted lists, tables and images can be created from R objects.

The functionality is extremely similar to what is currently available using Sweave.

To process the document in R, the basic syntax is:

```
odfWeave("someFile.odt", "outputFile.odt")
```

The process is virtually the same, although the resulting .odt file has the images embedded (in a zipped sub–directory).

# Creating Tables

The `odfTable` function serves the same purpose as the `latex` function: to create markup for tabular representations of data.

For example, the basic usage is

```
<<corrTable, echo=FALSE, results=xml>>=
   odfTable(pkgVersions("matrix"))
@
```

| | | |
|---|---|---|
| base (v.2.5.0) | lattice (v.0.15-5) | utils (v.2.5.0) |
| datasets (v.2.5.0) | methods (v.2.5.0) | XML (v.1.6-3) |
| graphics (v.2.5.0) | odfWeave (v.0.5.9) | |
| grDevices (v.2.5.0) | stats (v.2.5.0) | |

More complex tables, with captions, can also be generated.

| | Frost | Population | Area |
|---|---|---|---|
| Connecticut | 139 | 3100 | 4862 |
| Maryland | 101 | 4122 | 9891 |
| Pennsylvania | 126 | 11860 | 44966 |
| Virginia | 85 | 4981 | 39780 |

*Table 1: a table using differential formatting*

# Creating Images

Images are created using the code chunk options `fig = TRUE`. There are no `eps` and `pdf` options.

To choose the type and dimensions of the images, the `getImageDefs` and `setImageDefs` functions are used:

```
imageInfo <- getImageDefs()
imageInfo$plotHeight <- 3
imageInfo$plotWidth <- 4
setImageDefs(imageInfo)
```

The user can change the image type (almost anything but PDF and SVG), the device and the device options. The image dimensions and the size of the *displayed* image can be set independently. The function `adjustImageSize` provides a shortcut.

```
adjustImageSize(3, 4)
```

# Changing Format Definitions

There are style elements for tables (cells and text), bulleted lists, paragraphs and pages. Styles are changed in two steps

1. create/modify style *definitions*
2. declare the current styles to be used

Similar to images, styles can be set via the same type of get/set functions.

Style definitions must be created prior to running `odfWeave` via `setStyleDefs`:

```
styles <- getStyleDefs()
styles$smaller <- styles$ArialCentered
styles$smaller$fontSize <- "8 pt"
setStyleDefs(styles)
```

# Changing Format

Within the ODF document, the styles can be changed using code chunks:

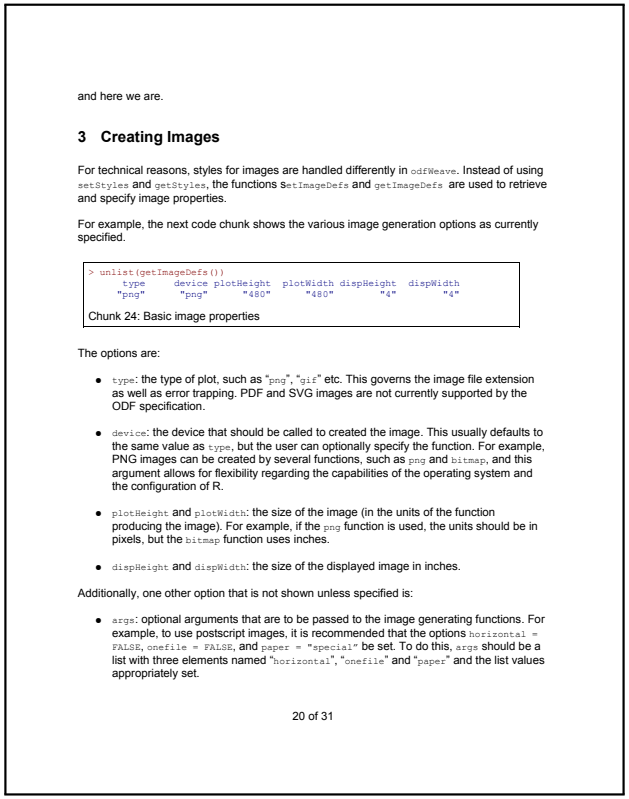
```
<<table1, echo = FALSE, results = xml>>=
odfTable(tableData)
@


<<table2, echo = FALSE, results = xml>>=
current <- getStyles()
current$cellText <- "smaller"
setStyles(current)

odfTable(tableData)
odfTableCaption("a table using differential formatting")
@
```

Page styles are a little different and are changed with the `odfSetPageStyle` and `odfPageBreak` functions.

# odfWeave Documentation

While there is a high degree of similarity between traditional Sweave tags and their analogs in odfWeave, differences do exist.

The examples directory of the package contains formatting.odt, which provides details about generating various objects using odfWeave.



and here we are.

### 3 Creating Images

For technical reasons, styles for images are handled differently in odfWeave. Instead of using setStyles and getStyles, the functions setImageDefs and getImageDefs are used to retrieve and specify image properties.

For example, the next code chunk shows the various image generation options as currently specified.

```
> unlist(getImageDefs())
      type     device plotHeight  plotWidth dispHeight  dispWidth
     "png"      "png"      "480"      "480"        "4"        "4"
```

Chunk 24: Basic image properties

The options are:

- type: the type of plot, such as "png", "gif" etc. This governs the image file extension as well as error trapping. PDF and SVG images are not currently supported by the ODF specification.
- device: the device that should be called to created the image. This usually defaults to the same value as type, but the user can optionally specify the function. For example, PNG images can be created by several functions, such as png and bitmap, and this argument allows for flexibility regarding the capabilities of the operating system and the configuration of R.
- plotHeight and plotWidth: the size of the image (in the units of the function producing the image). For example, if the png function is used, the units should be in pixels, but the bitmap function uses inches.
- dispHeight and dispWidth: the size of the displayed image in inches.

Additionally, one other option that is not shown unless specified is:

- args: optional arguments that are to be passed to the image generating functions. For example, to use postscript images, it is recommended that the options horizontal = FALSE, onefile = FALSE, and paper = "special" be set. To do this, args should be a list with three elements named "horizontal", "onefile" and "paper" and the list values appropriately set.

20 of 31

# Document Generation Process

In each Midas and isMG, we have an R module that extracts the relevant information and processes the ODF template to produce another ODF file with tables, figures etc.

The ODF file can be opened by OpenOffice, but we do an additional step to convert the document to the Microsoft doc format. OpenOffice has an API that can be used to do format conversions.

We use a bash script that does the conversion at the command line:

```
ooconvert afterWeaving.odt finalReport.doc
```

Other public code exists to use OpenOffice for the conversion:

sourceforge.net/projects/ooconvert/

www.artofsolving.com/opensource/jodconverter

# Tradeoffs

Pros:

- we can create editable reports
- no markup knowledge needed
- wider variety of output and image formats
- format conversion is reliable and not difficult
- images saved in natural format
- ODF is fairly feature–rich

Cons:

- ODF processing is not as efficient or straight–forward as LaTeX/Sweave
- ODF is new and more subject to change
- not as many R tools for ODF
- need an external zip program
- still may need to convert to other formats

# To Do List

- Tangling
- Compatibility with presentation (`.odp`) files
- Write ODF analogs to common print functions (e.g. `summary.lm`)
- Better table formatting
  - uneven columns widths
  - native data formats (instead of treating numbers as text via `format`)
  - multi–column support

# Acknowledgements

Thanks to

- Friedrich Leisch for developing `Sweave` and the R Core Team for their amazing work
- Nathan Coulter for expertise on grep and encodings, and for writing `ooconvert`
- Steve Weston for writing the new XML parser
- Benevolent Overlord David Potter